ARbitrary PRECision Computation Package (ARPREC)
Copyright (C) 2003-2012
==========

Revised 25 Oct 2012

To build this library, follow the steps below.
Some system specific notes are at the end of this file.

Build Instructions
==========

1. Run the configure script by typing

```
./configure
```

The script will attempt to automatically detect various system-dependent
variables used during compilation (such as the C++/fortran compiler,
compiler flags, and linker flags).

If you want to specify a particular C++ / F90 compiler and their flags,
you can set them as environmental variables. For example:

```
FC=ifc FCFLAGS="-O2 -FR" ./configure
```

Important variables are

```
CXX       C++ compiler to use
CXXFLAGS  C++ compiler flags to use
CC        C compiler to use (for C demo program)
CFLAGS    C compiler flags to use (for C demo program)
FC        Fortran 90 compiler
FCFLAGS   Fortran 90 compiler flags to use
FCLIBS    Fortran 90 libraries needed to to link with C++ code.
```

See ./configure --help to see other options.

3. The configure script should also have created the files 'config.h' and
'include/arprec/arprec_config.h', which will contain the compile time
defines. Examine these and edit them if necessary. In most cases
no edits are necessary, since the options are detected when configure
was run.

4. Type "make". This will build the library, and necessary Fortran
   wrappers.

5. Optionally, one can build and run some simple test programs.
   To do this, type "make check". Some programs run during this
   phase is a good demonstration of how to use the qd library in C+
+.

6. You can now install the ARPREC library by issuing "make install".

7. If you want to build some sample programs written in C++
   you can type "make cpp-demo".

8. If you want to build some sample programs written in Fortran 90,
   you can type "make fortran-demo".

9. If you want to compile the Experimental Mathematician's Toolkit,
   type "make toolkit". This will compile the Fortran-90 codes in
   the toolkit cirectory, including the "mathinit" and "mathtool"
   Read the "README" file in the toolkit directory for additional
details.

System-Specific Notes

Linux x86 / Itanium

You can use g++ to compile the C++ code. The Fortran 90 codes
can be compiled using Intel Fortran 95 compiler

  http://www.intel.com/software/products/compilers/flin/

available freely for non-commercial uses. There is also a C++
compiler available (for non-commercial use) at

  http://www.intel.com/software/products/compilers/clin/

which can be used to compile the C++ portion. By default the
configure script will use the Intel compiler if found.

Apple (OS X)

For Apple OS X Intel-badsed systems, it is recommended that you use
the g++ (version 4.0 or higher) C++ compiler and the gfortran Fortran
compiler. The g++ compiler and related command-line tools are now
available via this URL:

https://developer.apple.com/downloads/index.action

If you do not already have an Apple Developer account, you must register [free] -- typically use your iTunes account username (email) and password. Once you login, look for "Command Line Tools" for Mountain Lion or whatever version of the Apple OS you have installed. The Command Line Tools can also be installed if you have the full Xcode App installed -- look for option to download "Command Line Tools".

The gfortran compiler can be downloaded from:
http://gcc.gnu.org/wiki/GFortranBinaries#MacOS

After installing both of these packages, type "which g++" and "which gfortran". If you don't "see" them with these commands, you may need to add the directory where they reside, typically

/usr/bin/g++

(for g++) and

/usr/local/bin/gfortran

(for gfortran)

to your default path search, typically with a line such as

PATH=/usr/bin:/usr/local/bin:$PATH

inserted in the .bashrc file in your home directory.

When this has been done, in the main arprec directory type

```
./configure CXX=g++ FC=gfortran FCFLAGS=-m64
```

then type "make" to construct the library. See the "README" file on how to construct a compile-link script for your own codes.

IBM (Power)

With IBM's xlC/xlf90 compilers, you may want to experiment with --enable-fma option which uses a faster code but relies on the compiler to generate a fused multiply-accumulate instruction. WARNING: since the compiler is not required to produce such instructions, this is not guaranteed to work. Please test before using.